EAP
Internet Draft

H. Tschofenig
D. Kroeselberg
Siemens
Y. Ohba
Toshiba

Document: draft-tschofenig-eap-ikev2-03.txt
Expires: August 2004 February 2004

# EAP IKEv2 Method (EAP-IKEv2)

## Status of this Memo



## Abstract

EAP-IKEv2 is an EAP method which reuses the cryptography and the payloads of IKEv2, creating a flexible EAP method that supports both symmetric and asymmetric authentication. This EAP method offers the security benefits of IKEv2 authentication and key agreement without the goal of establishing IPsec security associations.

Table of Contents



## 1. Introduction

This document specifies the EAP-IKEv2 authentication method. The main design goal for EAP-IKEv2 is to provide a flexible and efficient EAP method which makes the IKEv2 protocol's features available for scenarios using EAP-based authentication.
The main advantage of EAP-IKEv2 is that it does not define a new cryptographic protocol, but re-uses the IKEv2 authentication exchanges, and thereby provides strong, well-analyzed, cryptographic properties as well as broad flexibility.

EAP-IKEv2 provides mutual authentication between EAP peers. This may be based on either symmetric methods using pre-shared keys, or on

asymmetric methods based on public/private key pairs, Certificates and CRLs. It is possible to use different types of authentication for the different directions, e.g. the server uses certificate-based authentication whereas the client uses a symmetric method. IKEv2 supports two-phased authentication schemes by establishing a server-authenticated secure tunnel and subsequently protecting an EAP authentication allowing for legacy client authentication





methods. EAP-IKEv2, however, does not support this optional tunneling feature of IKEv2 in this version, which allows to increase the EAP-IKEv2 method performance and decrease implementation complexity.

A non-goal of EAP-IKEv2 (and basically the major difference to plain IKEv2) is the establishment of IPsec security associations, as this would not make much sense in the standard AAA three-party scenario, consisting of an EAP peer, an authenticator (NAS) and a back-end authentication server terminating EAP. IPsec SA establishment may be required locally (i.e., between the EAP peer and some access server). However, SA establishment within an EAP method would only provide SAs between the EAP peer and the back-end authentication server. Other approaches as e.g., those of the IETF PANA group are considered more appropriate in this case.

## 2. IKEv2 and EAP-IKEv2 Overview

IKEv2 [Kau04] is a protocol which consists of two exchanges:

(1) an authentication and key exchange protocol which establishes an IKE-SA.

(2) messages and payloads which focus on the negotiation of parameters in order to establish IPsec security associations (i.e., Child-SAs). These payloads contain algorithm parameters and traffic selector fields.

In addition to the above-mentioned parts IKEv2 also includes some payloads and messages which allow configuration parameters to be exchanged primarily for remote access scenarios.

The EAP-IKEv2 method defined by this document uses the IKEv2 payloads and messages used for the initial IKEv2 exchange which

establishes an IKE-SA.

IKEv2 provides an improvement over IKEv1 [RFC2409] as described in Appendix A of [Kau04]. Important for this document are the reduced number of initial exchanges, decreased latency of the initial exchange, and some other fixes (e.g., hash problem). IKEv2 is a cryptographically sound protocol that has received a considerable amount of expert review and that benefits from a long practical experience with IKE.
The goal of EAP-IKEv2 is to inherit these properties within an efficient, secure EAP method.

In addition, IKEv2 provides authentication and key exchange capabilities which allow an entity to use symmetric as well as





asymmetric authentication within a single protocol. Such flexibility is considered important for an EAP method and is provided by EAP-IKEv2.

[Per03] provides a good tutorial for IKEv2 design decisions.

EAP-IKEv2 provides a secure fragmentation mechanism in which integrity protection is performed for each fragment of an IKEv2 message.

## 3. Terminology

This document does not introduce new terms other than those defined in [RFC2284] or in [Kau04].

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

## 4. Protocol overview

This section provides some overview over EAP-IKEv2 message exchanges. Note that some mandatory IKEv2 payloads are omitted, or profiled (such as SAi2 and SAr2), as it is not supported to establish IPsec (ESP, AH) SAs in EAP-IKEv2.

IKEv2 uses the same protocol message exchanges for both symmetric and asymmetric authentication. The difference lies only in the computation of the AUTH payload. See Section 2.15 of [Kau04] for more information about the details of the AUTH payload computation. It is even possible to combine symmetric (e.g., from the client to the server) with asymmetric authentication (e.g., from the server to the client) in a single protocol exchange. Figure 1 depicts such a protocol exchange.

Message exchanges are reused from [Kau04], and are adapted. Since this document does not describe frameworks or particular architectures the message exchange takes place between two parties - between the Initiator (I) and the Responder (R). In the context of EAP the Initiator takes the role of the EAP server and the responder matches the EAP peer.

The first message flow shows the EAP-IKEv2 full successful exchange. The core EAP-IKEv2 exchange (message (3) - (6)) consists of four messages (two round trips)_only. The first two messages constitute the standard EAP identity exchange and are optional; they are not required in case the EAP server is known. In the exchange, the EAP





server (B) takes the role of the IKEv2 initiator and the EAP peer (A) acts as the IKEv2 responder.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH})

7) A <-- B: EAP-Success
```

Figure 1: EAP-IKEv2 successful message flow

Figure 2 shows the message flow in case the EAP peer fails to authenticate the EAP server.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

7) A <-- B: EAP-Failure
```

Figure 2: EAP-IKEv2 with failed server authentication

Figure 3 shows the message flow in case the EAP server fails to authenticate the EAP peer. The EAP peer MUST send an empty EAP-IKEv2





informational message in reply to the EAP server's error indication. The EAP server answers with an EAP-Failure.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})
```

```
6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH})

7) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

9) A <-- B: EAP-Failure
```

Figure 3: EAP-IKEv2 with failed client authentication

Since the goal of this EAP method is not to establish an IPsec SA some payloads used in IKEv2 are omitted. In particularly the following messages and payloads SHOULD not be sent:

- Traffic Selector (TS)payloads
- SA payloads that carry SA proposals for protocol IDs other than 1(IKE), i.e., SA payloads with protocol ID 2 (ESP) or 3 (AH)
- ESN (extended sequence number) transforms

Some of these messages and payloads are optional in IKEv2. In general it does not make sense to directly negotiate IPsec SAs within EAP-IKEv2, as such SAs are not required between the EAP endpoints and as SAs cannot be transferred to different AAA entities by standard AAA protocols.

Consequently, mechanisms and payloads that are not supported by EAP-IKEv2 are:
- ECN Notifications as specified in section 2.24 of [Kau04].
- IKE-specific port handling
- NAT traversal





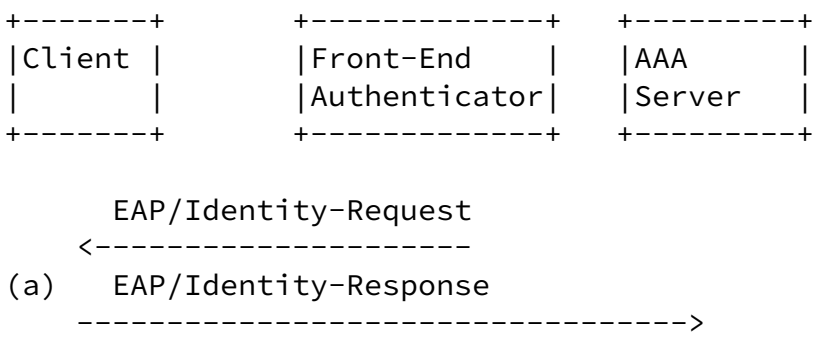
Since the EAP server acts as the initiator of the initial IKEv2 exchange, a number of optional payloads used for realizing specific features in IKEv2 are not supported by EAP-IKEv2, as they are intended for the client side (e.g. for corporate access scenarios) in plain IKEv2. These payloads MUST not be sent by an EAP-IKEv2 entity. EAP-IKEv2 entities receiving such payloads MUST respond with

the appropriate error messages as defined in [Kau04]. These payloads are:
- Configuration (CFG) payloads as specified in 3.15 of [Kau04]. These payloads MUST not be sent by an EAP-IKEv2 implementation. EAP-IKEv2 entities receiving such payloads MUST ignore configuration payloads as described for minimal implementations in 3.15 of [Kau04].
- EAP payloads as specified in section 3.16 of [Kau04]. These payloads allow to run an inner EAP exchange for secure legacy authentication through an IKE SA. EAP-IKEv2 implementations acting as initiator MUST include and AUTH payload in the initial IKE_AUTH message (message 3 of the initial IKE exchange). EAP-IKEv2 implementations receiving initial IKE_AUTH messages as responders that indicate the initiator's desire to start extended authentication MUST be answered with an AUTHENTICATION_FAILED notification as the response.

IKEv2 provides optional functionality for additional DoS protection by adding a roundtrip to the initial exchanges, see section 2.xx of [Kau04]. As this is intended to protect the IKEv2 responder but in EAP-IKEv2 the EAP server takes the role of the initiator, it is not recommended to use this feature of IKEv2 for server protection.

## 5. Identities used in EAP-IKEv2

A number of different places allow to convey identity information in IKEv2, when combined with EAP. This section describes their function within the different exchanges of EAP-IKEv2. Note that EAP-IKEv2 does not introduce more identities than other non-tunneling EAP methods. Figure 4 shows which identities are used during the individual phases of the protocol.



```
        Tunnel-Establishment
 (b)    (Identities of IKEv2 are used)
        Server (Network) Authentication
     <-----------------------------------
                    ...
     ----------------------------------->
```

Figure 4: Identities used in EAP-IKEv2

a) The first part of the (outer) EAP message exchange provides information about the identities of the EAP endpoints. This message exchange mainly is an identity request/response. This exchange is optional if the EAP server is known already or can be learned by other means.

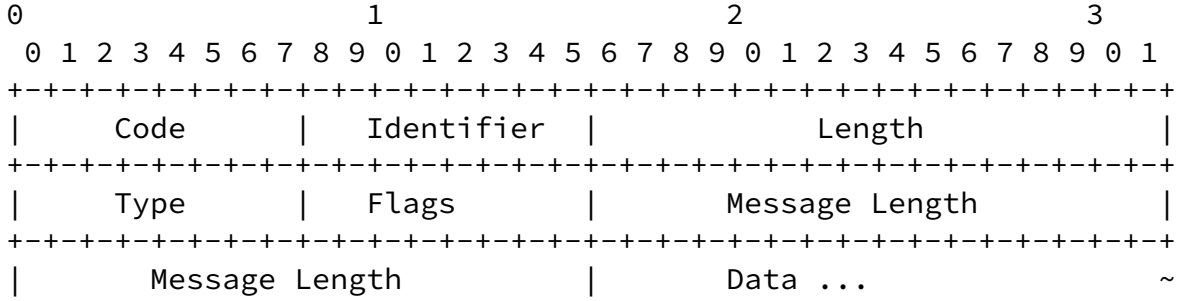
b) Identities exchanged within EAP-IKEv2 for both the initiator and the responder. The initiator identity is often associated with a user identity such as a fully-qualified RFC 822 email address. The identity of the responder might be a FQDN. The identity is of importance for authorization.

For carrying identities in EAP-IKEv2, implementations MUST follow the rules given in [Kau04], section 3.5, i.e., MUST be configurable to send at least one of ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, or ID_KEY_ID, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types.

## 6. Packet Format

The IKEv2 payloads, which are defined in [Kau04], are embedded into the Data field of the standard EAP Request/Response packets. The Code, Identifier, Length and Type field is described in [RFC2284]. The Type-Data field carries a one byte Flags field following the IKEv2 payloads. Each IKEv2 payload starts with a header field HDR (see [Kau04]).

The packet format is shown in Figure 5.



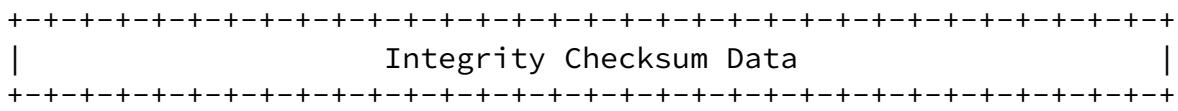



Figure 5: Packet Format

No additional packet formats other than those defined in [Kau04] are required for this EAP method.

The Flags field is used for fragmentation support. The S and F bits are reserved for future use.

Currently five bits of the eight bit flags field are defined. The remaining bits are set to zero.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S F L M I 0 0 0|
+-+-+-+-+-+-+-+-+
```

S = (reserved)
F = (reserved)
L = Length included
M = More fragments
I = Integrity Checksum Data included

EAP-IKEv2 messages which have neither the S nor the F flag set contain regular IKEv2 message payloads inside the Data field.

With regard to fragmentation we follow the suggestions and descriptions given in Section 2.8 of [PS+03]: The L indicates that a length field is present and the M flag indicates fragments. The L flag MUST be set for the first fragment and the M flag MUST be set on all fragments expect for the last one. Each fragment sent must subsequently be acknowledged.

The Message Length field is four octets long and present only if the L bit is set. This field provides the total message length that is being fragmented (i.e., the length of the Data field.).

The Integrity Checksum Data is the cryptographic checksum of the entire EAP message starting with the Code field through the Data

field. This field presents only if the I bit is set. The field immediately follows the Data field without adding any padding octet before or after itself. The checksum MUST be computed for each fragment (including the case where the entire IKEv2 message is carried in a single fragment) by using the same key (i.e., SK_ai or SK_ar) that is used for computing the checksum for the IKEv2





Encrypted payload in the encapsulated IKEv2 message. The Integrity Checksum Data field is omitted for other packets. To minimize DoS attacks on fragmented packets, messages that are not protected SHOULD NOT be fragmented. Note that IKE_SA_INIT messages are the only ones that are not encrypted or integrity protected, however, such messages are not likely to be fragmented since they do not carry certificates.

The EAP Type for this EAP method is <TBD>.

## 7. Retransmission

Since EAP authenticators support a timer-based retransmission mechanism for EAP Requests and EAP peers retransmit the last valid EAP Response when receiving a duplicate EAP Request message, IKEv2 messages MUST NOT be retransmitted based on timers, when used as EAP authentication method.

## 8. Key derivation

The EAP-IKEv2 method described in this document generates session keys. On the one hand, these session keys are used within the IKE-SA, for protection of EAP-IKEv2 payloads, e.g., AUTH exchanges or notifications. On the other hand, additional keys are derived to be exported as part of the EAP keying framework [AS+03] (i.e., MSK, EMSK and IV). It is good cryptographic security practice to use different keys for different "applications". Hence we suggest reusing of the key derivation function suggested in Section 2.17 of [Kau04] to create keying material KEYMAT.

The key derivation function defined is KEYMAT = prf+(SK_d, Ni | Nr), where Ni and Nr are the Nonces from the IKE_SA_INIT exchange.

Since the required amount of keying material is greater than the

size of the output of the prf algorithm the prf is used iteratively. Section 2.13 of [Kau04] describes this mechanism in detail.

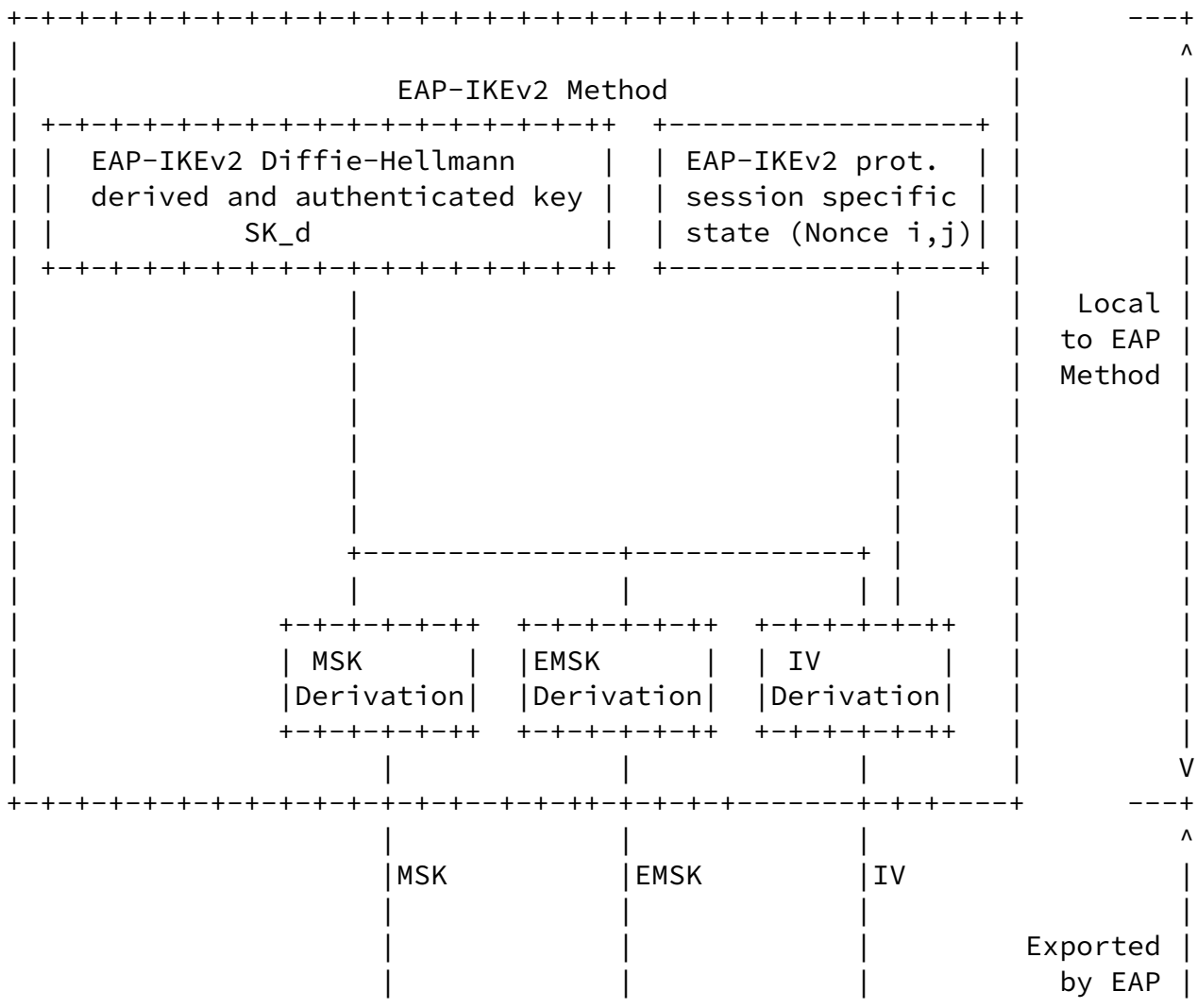
According to [AS+03] the keying material of MSK, EMSK and IV have to be at minimum 64, 64 and 64 octets long.

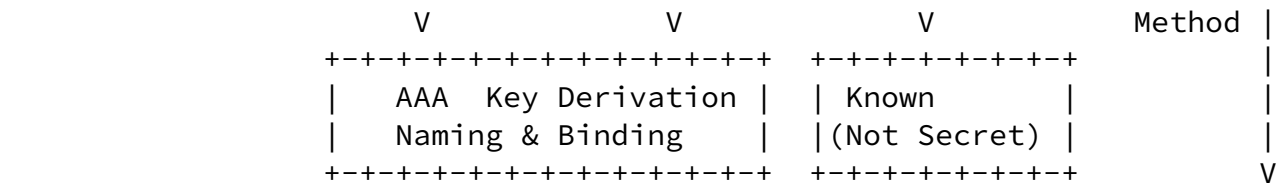
The produced keying material for MSK, EMSK and IV MUST be at least the minimum size (i.e., 64 octets). The keying material KEYMAT is split into the MSK, EMSK and IV part.

Figure 6 describes the keying hierarchy of EAP-IKEv2 graphically. This figure is adopted from Figure 2 of [AS+03].










Legend:

MSK = Master Session Key (512 Bit)
EMSK = Extended Master Session Key (512 Bit)
SK_d = Session key derived by EAP-IKEv2
IV = Initialization Vector

Figure 6: EAP-IKEv2 Keying Hierarchy

## 9. Error Handling

As described in the IKEv2 specification, there are many kinds of errors that can occur during IKE processing (i.e., processing the





Data field of EAP-IKEv2 Request and Response messages) and detailed processing rules. EAP-IKEv2 follows the error handling rules specified in the IKEv2 specification for errors on the Data field of EAP-IKEv2 messages, with the following additional rules:

For an IKEv2 error that triggers an initiation of an IKEv2 exchange (i.e., an INFORMATIONAL exchange), an EAP-IKEv2 message that contains the IKEv2 request that is generated for the IKEv2 exchange MUST be sent to the peering entity. In this case, the EAP message that caused the IKEv2 error MUST be treated as a valid EAP message.

For an IKEv2 error for which the IKEv2 message that caused the error is discarded without triggering an initiation of an IKEv2 exchange, the EAP message that carries the erroneous IKEv2 message MUST be treated as an invalid EAP message and discarded as if it were not received at EAP layer.

For an error occurred outside the Data field of EAP-IKEv2 messages, including defragmentation failures, integrity check failures, errors in Flag and Message Length fields, the EAP message that caused the error MUST be treated as an invalid EAP message and discarded as if it were not received at EAP layer.

When the EAP-IKEv2 method runs on a backend EAP server, an outstanding EAP Request is not retransmitted based on timer and thus there is a possibility of EAP conversation stall when the EAP server receives an invalid EAP Response. To avoid this, the EAP server MAY retransmit the outstanding EAP Request in response to an invalid EAP Response. Alternatively, the EAP server MAY send a new EAP Request in response to an invalid EAP Response with assigning a new Identifier and putting the last transmitted IKEv2 message in the Data field.

## 10. Fast Reconnect

EAP-IKEv2 supports fast reconnect, i.e., a successful reconnect exchange creates a new IKE-SA by using an IKE CHILD_SA exchange. The purpose of a re-authentication exchange is to allow for efficient re-keying based on the existing IKE-SA in situations where (depending on the given security policy) no full authentication is required in case of an existing EAP-IKEv2 security context.
The fast reconnect exchange uses the IKE-SA rekeying as specified in section 2.18 of [IKEv2]. However, the exchanges for EAP-IKEv2 do not use the payloads for rekeying other than IKE SAs:
- The TS (traffic selector) payloads SHOULD not be sent by EAP-IKEv2 implementations.
- The [N] payload (REKEY_SA notification) SHOULD not be sent by EAP-IKEv2 implementations.





During fast re-authentication, the new IKE_SA is computed as specified in [IKEv2], section 2.18. The new keying material derived from this IKE_SA is computed as in an initial EAP-IKEv2 authentication exchange.
Fast re-authentication allows for an optional new Diffie-Hellman exchange.

The following exchange provides fast reconnect for EAP-IKEv2, where A is the EAP peer (IKE responder) and B is the EAP server (IKE initiator):

1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

```
3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Nr, [KEr]})

5) A <-- B: EAP-Success
```

Figure 7: Fast Reconnect Message Flow

The first two messages constitute the standard EAP identity exchange and are optional; they are not required in case the EAP server is known.

Figure 8 shows the fast reconnect message flow in case the EAP peer fails to re-authenticate the EAP server.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2
         (HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
            HDR, SK {N(AUTHENTICATION_FAILED)})

5) A <-- B: EAP-Failure
```

Figure 8: EAP-IKEv2 fast reconnect (server authentication failed)





Figure 9 shows the fast reconnect message flow in case the EAP server fails to re-authenticate the EAP peer. The EAP peer MUST send an empty EAP-IKEv2 informational message in reply to the EAP server's error indication. The EAP server answers with an EAP-Failure.

```
1) A <-- B: EAP-Request/Identity
```

```
2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Nr, [KEr]})

5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

7) A <-- B: EAP-Failure
```

Figure 9: EAP-IKEv2 fast reconnect (client authentication failed)

IKE_SAs do not have lifetimes. Such lifetimes are therefore set by local policies of the peers. Typically the peer setting the shorter lifetime will therefore trigger the reconnect procedure.

Note: IKEv2 supports fast rekeying to be initiated by both peers. As the EAP server initiating the rekeying better fits the EAP message flow, the EAP server, from an IKE perspective, is the initiator in the rekeying exchange.

## 11. Channel Binding

EAP-IKEv2 provides a channel binding functionality [RFC2284bis] in order for the EAP peer and EAP server to make sure that the both entities are given the same network access attributes such as Calling-Station-Id, Called-Station-Id, and NAS-Port-Type by the NAS. This is achieved by using Notify payloads to exchange attribute data between the EAP peer and EAP server.





A Notify payload that carries a null channel binding attribute is

referred to as a channel binding request. A Notify payload which contains a non-null channel binding attribute and is sent in response to a channel binding request is referred to as a channel binding response. A pair of channel binding request and channel binding response constitute a channel binding exchange. A distinct Notify payload type is used for a particular type of channel binding attribute, which is referred to as a channel binding attribute type. It is allowed to carry multiple channel binding requests and/or responses with different channel binding attribute types in a single IKEv2 message. A set of channel binding exchanges performed in a single round of EAP-IKEv2 full authentication or fast reconnect is referred to as a channel binding procedure.

A Notify payload that is used for reporting an error occurred during a channel binding exchange is referred to as a channel binding error indication.

EAP-IKEv2 offers a protected result indication mechanism (see section 12.2). To receive protected result indication, the EAP server MUST initiate a channel binding exchange as specified in Figure 10, message 5. As a result of this channel binding exchange, the client will receive a protected result indication, because the server will initiate an informational exchange as part of the channel binding procedure(messages 7 and 8) through the new IKE-SA that results from a successful reconnect procedure.

## 11.1 Channel Binding Procedure in Full Authentication

In the case of EAP-IKEv2 full authentication procedure, the channel binding procedure is performed in the following way.

The EAP peer MAY include one or more channel binding request in an IKE_SA_INIT response. The EAP server MAY include one or more channel binding request in an IKE_AUTH request. When the EAP server receives an IKE_SA_INIT response with one or more channel binding request, it MUST include the corresponding channel binding response(s) an IKE_AUTH request (in addition to its channel binding request(s) if any). When the EAP peer receives an IKE_AUTH request with one or more channel binding request, it MUST include the corresponding channel binding response(s) in an IKE_AUTH response.

When the EAP server successfully validates all the channel binding response(s) sent by the EAP server, it initiates an INFORMATIONAL exchange, where an empty payload is used in both INFORMATIONAL request and INFORMATIONAL response. This exchange serves as a protected success indication. After completion of this INFORMATIONAL exchange, the EAP server sends Success message.

## 11.2 Channel Binding Procedure in Fast Reconnect

In the case of EAP-IKEv2 fast reconnect, the channel binding procedure is performed in the following way.

In the pair of CREATE_CHILD_SA exchange, the EAP peer and/or the EAP server MAY include one or more channel binding request, one for each channel binding attribute that needs validation. When the EAP peer receives a CREATE_CHILD_SA request with containing one or more channel binding request, it MUST contain channel binding response(s) in the CREATE_CHILD_SA response, as well as its channel binding request(s) if any. This piggybacking is possible because the CREATE_CHILD_SA exchange is protected with the old IKE_SA. When the EAP server receives a CREATE_CHILD_SA response, if it has one or more channel binding response to send to the EAP peer, it initiates an INFORMATIONAL exchange immediately after completion of the CREATE_CHILD_SA exchange, where one or more channel binding response is carried in the INFORMATIONAL request. If the EAP peer successfully validates the channel binding response(s), it MUST respond with an empty INFORMATIONAL response. This exchange serves as a protected success indication. After completion of this INFORMATIONAL exchange, the EAP server sends Success message.

## 11.3 Channel Binding Error Indication

A channel binding error is detected by the EAP peer or EAP server when (i) a channel binding response is not contained in the expected IKEv2 message or (ii) a channel binding response is contained in the expected IKEv2 message but the channel binding attribute does not have the expected value. Whenever a channel binding error is detected, the detecting entity MUST send a channel binding error indication to its peering entity. In case of (ii), the channel binding error indication MUST contain the channel binding attribute that caused the error.

When the EAP server detects a channel binding error, a channel binding error indication MUST be carried in an INFORMATIONAL request, and the EAP peer MUST respond with an empty INFORMATIONAL response.
When the EAP peer detects a channel binding error, a channel binding error indication MUST be carried in an IKEv2 error reporting message for which the R-flag of the IKEv2 header MUST be set. The EAP server MUST respond with EAP-Failure message when it receives such a channel binding error indication.

## 11.4 Notify Payload Types for Channel Binding

The following Notify Payload types are defined for the purpose of channel binding exchange.





CALLING_STATION_ID TBD
The payload data in a channel binding response of this type contains octet string representation of Calling-Station-Id value known to the EAP server by using an external mechanism.

CALLED_STATION_ID TBD
The payload data in a channel binding response of this type contains octet string representation of Called-Station-Id value known to the EAP peer by using an external mechanism.

NAS_PORT_TYPE TBD
The payload data in a channel binding response of this type contains 4-octet unsigned integer value of NAS-Port-Type known to the EAP peer by using an external mechanism.

The following Notify Payload types are defined for the purpose of reporting when there is an error in a channel binding exchange.

INVALID_CALLING_STATION_ID TBD

The payload data (if non-null) contains octet string representation of Calling-Station-Id value that caused the error.

INVALID_CALLED_STATION_ID TBD

The payload data (if non-null) contains octet string representation of Called-Station-Id value that caused the error.

INVALID_NAS_PORT_TYPE TBD

The payload data (if non-null) contains 4-octet unsigned integer value of NAS-Port-Type that caused the error.

Table 1 shows the the entity that is allowed to send a channel

binding request for each channel binding attribute type.

| channel binding attribute type | The entity that is allowed to send channel binding request |
|---|---|
| CALLING_STATION_ID | EAP server |
| CALLED_STATION_ID | EAP peer |
| NAS_PORT_TYPE | EAP server |





Table 1: Channel Binding Attribute Types and Requesting Entities

## 11.5 Examples

In the figures of this section, a Notify payload tagged with '*' indicates a Notify payload with null data (i.e., a channel binding request). a Notify payload no tagged with '*' indicates a Notify payload with non-null data (i.e., a channel binding response).

Figure 10 shows an example of EAP-IKEv2 authentication sequence with a successful channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
         N(CALLED_STATION_ID*))

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
         N(CALLED_STATION_ID),
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
```

```
          HDR(A,B), SK {IDr, [CERT,] AUTH,
          N(CALLING_STATION_ID),
          N(NAS_PORT_TYPE)})

   7) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {})

   8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {})

   9) A <-- B: EAP-Success
```

Figure 10: EAP-IKEv2 with successful channel binding

Figure 11 shows an example of EAP-IKEv2 authentication sequence when the EAP server detects an error in a channel binding procedure. The





first two messages constitute the standard EAP identity exchange and are optional. In this case, message 7) and 8) MUST constitute an INFORMATIONAL exchange.

```
   1) A <-- B: EAP-Request/Identity

   2) A --> B: EAP-Response/Identity(Id)

   3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

   4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
          N(CALLED_STATION_ID*))

   5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
          N(CALLED_STATION_ID),
          N(CALLING_STATION_ID*),
          N(NAS_PORT_TYPE*)})

   6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {IDr, [CERT,] AUTH,
          N(CALLING_STATION_ID),
          N(NAS_PORT_TYPE)})
```

```
7) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLING_STATION_ID)})

8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

9) A <-- B: EAP-Failure
```

Figure 11: EAP-IKEv2 with channel binding error (detected by EAP server)

Figure 12 shows an example of EAP-IKEv2 authentication sequence when the EAP peer detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
```





```
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
         N(CALLED_STATION_ID*))

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
         N(CALLED_STATION_ID),
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLED_STATION_ID)})

7) A <-- B: EAP-Failure
```

Figure 12: EAP-IKEv2 with channel binding error (detected by EAP peer)

Figure 13 shows an example of EAP-IKEv2 fast reconnection sequence with a successful channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
         N(CALLED_STATION_ID*),
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(CALLED_STATION_ID)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR(A,B), SK {})

7) A <-- B: EAP-Success
```

Figure 13: Fast reconnect with channel binding error (fast reconnect)

Figure 14 shows an example of EAP-IKEv2 fast reconnect sequence when the EAP server detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and





are optional. In this case, message 7) and 8) MUST constitute an INFORMATIONAL exchange.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})
```

```
4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
         N(CALLED_STATION_ID*),
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLING_STATION_ID)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

7) A <-- B: EAP-Failure
```

Figure 14: Fast reconnect with channel binding error (detected by EAP server)

Figure 15 shows an example of EAP-IKEv2 fast reconnect sequence when the EAP peer detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
         N(CALLED_STATION_ID*),
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(CALLED_STATION_ID)})
```





```
6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLED_STATION_ID)})

7) A <-- B: EAP-Failure
```

Figure 15: Fast reconnect with channel binding error
(detected by EAP peer)

## 12. Security Considerations

### 12.1 General Considerations

The security of the proposed EAP method is intentionally based on IKEv2 [Kau04]. Therefore, the security claims of EAP-IKEv2 are derived from the security offered by the supported features of IKEv2.

### 12.2 Security Claims

Authentication mechanism:
Mutual authentication is supported based on either pre-shared symmetric keys or public/private key pairs. Besides certificates, plain public keys can be used. It is possible to use different types of authentication for the different directions within one authentication exchange. An example is the server using certificate-based authentication and the client using pre-shared secrets.

Password-based authentication should only be used in IKEv2 with extended authentication (EAP tunneling), which is not supported by this version of EAP-IKEv2. Therefore, dictionary attacks are not applicable in the context of EAP-IKEv2.

Man-in-the-middle attacks discovered in the context of tunneled authentication protocols (see [AN03] and [PL+03]) are not applicable to EAP-IKEv2 as the extended authentication feature of IKEv2 is not supported. Hence, the cryptographic binding claim is not applicable.

Ciphersuite negotiation is supported as specified in IKEv2 for IKE-SAs. The negotiation for IPsec (Child) SAs is not supported, as such SAs are not generated by EAP-IKEv2.

Protected result indication as described in section 7.16 of [RFC2284bis] is optionally provided by EAP-IKEv2. In message 5 of figure 1 (full successful authentication) the EAP server authenticates to the client. Message 6 authenticates the client to the server, and the client by authenticating the server and by sending message 6 expresses that it is willing to accept access. The client, however, does not get a protected result indication from the

server in this case. An attacker could potentially forge an EAP success/failure message which could result in DoS to the client. In some situations, synchronization may be achieved by lower layer indications.

Protected result indication is optionally provided as specified in section 11.
If this mechanism is not used, the recommended behavior for the client is to assume the correct establishment of a new IKE-SA after sending message 6, independent of the receipt of an EAP success/failure. In case of unsuccessful authentication, the server would answer with an IKEv2 notification (which, in case of the fast reconnect exchange, would be protected by the old IKE-SA). In case of a lost message 6, the server would retransmit message 5, indicating the message loss to the client.
The client implementation can minimize potential DoS risks due to missing protected result indications by assuming the correct establishment of a new IKE-SA after not receiving one of the above messages within a certain time window after sending message 6. In the fast reconnect case, the client needs to hold both the old and the new IKE-SA in parallel during this time window.

Session independence is optionally provided if the fast reconnect exchange includes the KE payloads (new Diffie-Hellman) as described in section 10, Figure 7.

Security claims:

| | |
|---|---|
| Ciphersuite negotiation: | Yes |
| Mutual authentication: | Yes |
| Integrity protection: | Yes |
| Replay protection: | Yes |
| Confidentiality: | Yes |
| Key derivation: | Yes |
| Key strength: | N/A |
| Dictionary attack prot.: | N/A |
| Fast reconnect: | Yes |
| Crypt. binding: | N/A |
| Protected result ind.: | yes |
| Session independence: | yes |
| Fragmentation: | Yes |
| Channel binding: | Yes |

## 13. Open Issues

The following issues are still under consideration:

- Notifications





IKEv2 provides the concept of notifications to exchange messages at any time (e.g., dead peer detection). It remains for further study which of these messages are required for this EAP method.

- supported identities

Can the NAI be carried by the RFC822 ID type of IKEv2? Are there other formats to be supported? Additional profiling may be required in section 5.

- protected result indication

This method provides protected result indications as an optional feature by running a channel binding exchange. It is ffs whether this feature should be mandated for all message flows (which would require to add an additional informational exchange to the message flows without channel binding.

- tunneled method

To reduce the method's complexity, EAP tunneling through EAP-IKEv2 that is in principal possible with IKEv2 is not supported. If tunneling support is, however, required (e.g. for sequencing), it is possible to develop an EAP-IKEv2-tunneled method from the present one. The major change would be to reverse the roles of IKEv2 initiator and responder, as the initiator is EAP-authenticated in the tunneled case.
It is not considered a good approach by the authors to have both the tunneled and the non-tunneled method in a single specification, as this would result in a rather complex method description. The tunneled-method EAP-IKEv2 specification, if required, will therefore come with a separate document.

## 14. Normative References

[RFC2284] L. Blunk and J. Vollbrecht: "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[Kau04] C. Kaufman: "Internet Key Exchange (IKEv2) Protocol", internet draft, Internet Engineering Task Force, January 2004. Work in progress.

[RFC2119] S. Bradner: "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Internet Engineering Task Force, March 1997.





## 15. Informative References

[AN03] N. Asokan, V. Niemi, and K. Nyberg: "Man-in-the-middle in tunnelled authentication", In the Proceedings of the 11th International Workshop on Security Protocols, Cambridge, UK, April 2003. To be published in the Springer-Verlag LNCS series.

[PL+03] J. Puthenkulam, V. Lortz, A. Palekar, D. Simon, and B. Aboba, "The compound authentication binding problem," internet draft, Internet Engineering Task Force, 2003. Work in progress.

[RFC2409] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[Per03] R. Perlman: "Understanding IKEv2: Tutorial, and rationale for decisions", internet draft, Internet Engineering Task Force, 2003. Work in progress.

[AS+03] B. Aboba, D. Simon and J. Arkko: " EAP Key Management Framework", internet draft, Internet Engineering Task Force, October, 2003. Work in progress.

[HS03] H. Haverinen, J. Salowey: "EAP SIM Authentication", internet draft, Internet Engineering Task Force, 2003. Work in progress.

[PS+03] A. Palekar, D. Simon, G. Zorn and S. Josefsson: "Protected EAP Protocol (PEAP)", internet draft, Internet Engineering Task Force, March 2003. Work in progress.

[AH03] J. Arkko and H. Haverinen: "EAP AKA Authentication", internet draft, Internet Engineering Task Force, June 2003. Work in progress.

Acknowledgments

We would like to thank Bernard Aboba, Jari Arkko, Guenther Horn, Paoulo Pagliusi and John Vollbrecht for their comments to this draft.

Additionally we would like to thank members of the PANA design team (namely D. Forsberg and A. Yegin) for their comments and input to the initial version of the draft.

Finally we would like to thank the members of the EAP keying design team for their discussion in the area of the EAP Key Management Framework.





Author's Addresses

Hannes Tschofenig
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany
EMail: Hannes.Tschofenig@siemens.com

Dirk Kroeselberg
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany
EMail: Dirk.Kroeselberg@siemens.com

Yoshihiro Ohba
Toshiba America Information Systems, Inc.
9740 Irvine Blvd.
Irvine, CA 92619-1697
USA
Phone: +1 973 829 5174
EMail: yohba@tari.toshiba.com

Full Copyright Statement

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.